In a Binary Search Tree (BST), the search process for a key follows a specific path determined by the properties of the tree. When searching for a key, you start at the root and decide whether to move left or right based on comparisons with the current node's key. If the search key is less than the current node's key, you move to the left child; if it is greater, you move to the right child. This process continues until the key is found or a leaf node is reached.

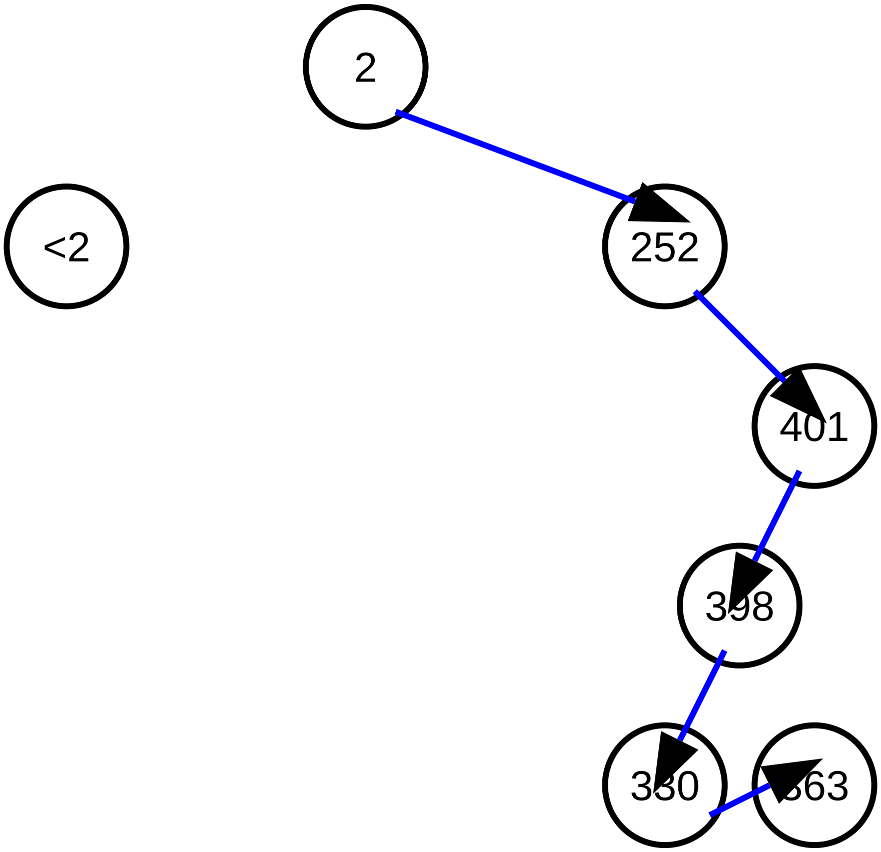
To determine which sequence cannot be the path followed when searching for the key 363, we need to check each sequence for violations of the BST properties:

In [6]:

```
from IPython.display import SVG
def show_svg(string):
    return SVG(string)


show_svg("test.svg")
```

Out[6]:



Sequence A: `2 252 401 398 330 363`

- 2 < 252 (valid move to the right)
- 252 < 401 (valid move to the right)
- 401 > 398 (valid move to the left)
- 398 > 330 (valid move to the left)

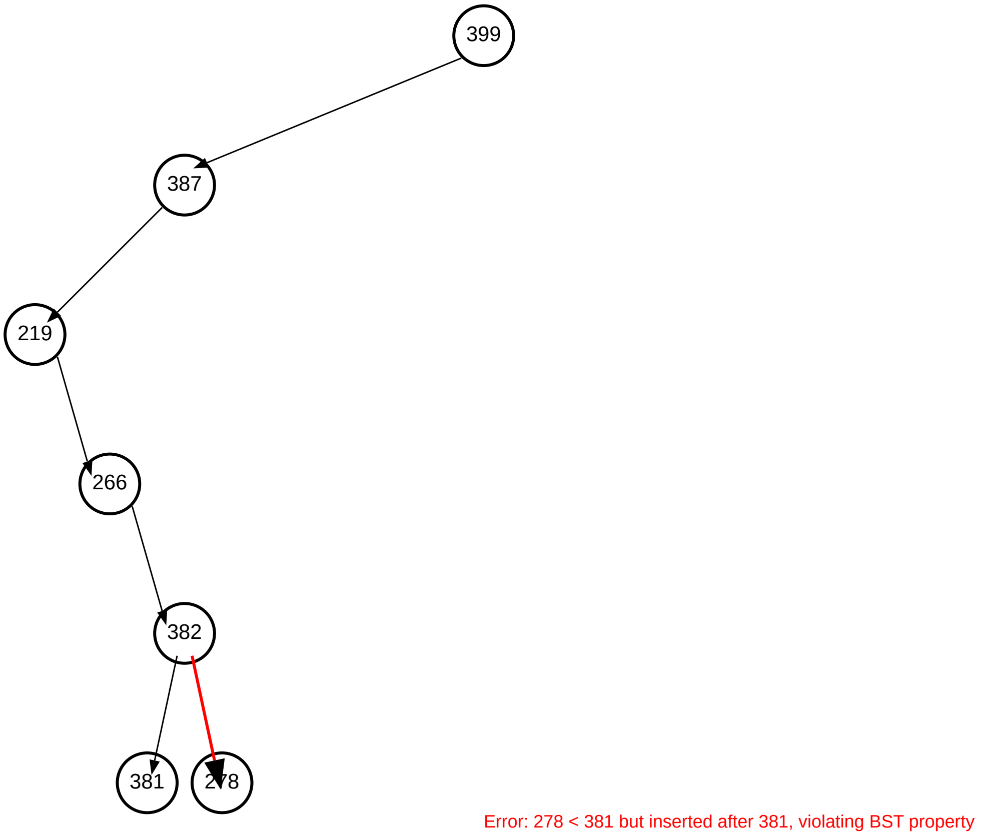
- 330 < 363 (valid move to the right)

This sequence is valid in a BST.

In [7]:

```
show_svg("t2.svg")
```

Out[7]:


399
387
219
266
382
381
278
Error: 278 < 381 but inserted after 381, violating BST property

Sequence B: `399 387 219 266 382 381 278 363`

Inserting the keys into a BST:

1. **Start with 399**:
   - 399 is the root.
2. **Insert 387**:
   - 387 < 399, so 387 is inserted to the left of 399.
3. **Insert 219**:
   - 219 < 399 (move left) and 219 < 387, so 219 is inserted to the left of 387.
4. **Insert 266**:
   - 266 < 399 (move left), 266 < 387 (move left), and 266 > 219, so 266 is inserted to the right of 219.
5. **Insert 382**:

  - 382 < 399 (move left), 382 < 387 (move left), 382 > 219 (move right), and 382 > 266, so 382 is inserted to the right of 266.

6. **Insert 381**:

   - 381 < 399 (move left), 381 < 387 (move left), 381 > 219 (move right), 381 > 266 (move right), and 381 < 382, so 381 is inserted to the left of 382.

7. **Insert 278**:

   - 278 < 399 (move left), 278 < 387 (move left), 278 > 219 (move right), 278 > 266 (move right), and 278 < 382.

   - But 278 should be inserted in the subtree rooted at 381. Since 278 is less than 381, it should go to the left of 381, but this violates the BST property because we've already placed 381 as the left child of 382, and any new left child should be less than 381.

   - **This is where the problem occurs**: **278 < 381** but would be inserted after 381 in the search, which is invalid in a BST structure.

8. **Insert 363**:

   - The process should continue to insert 363, but the tree structure is already invalid.

Key Takeaway:

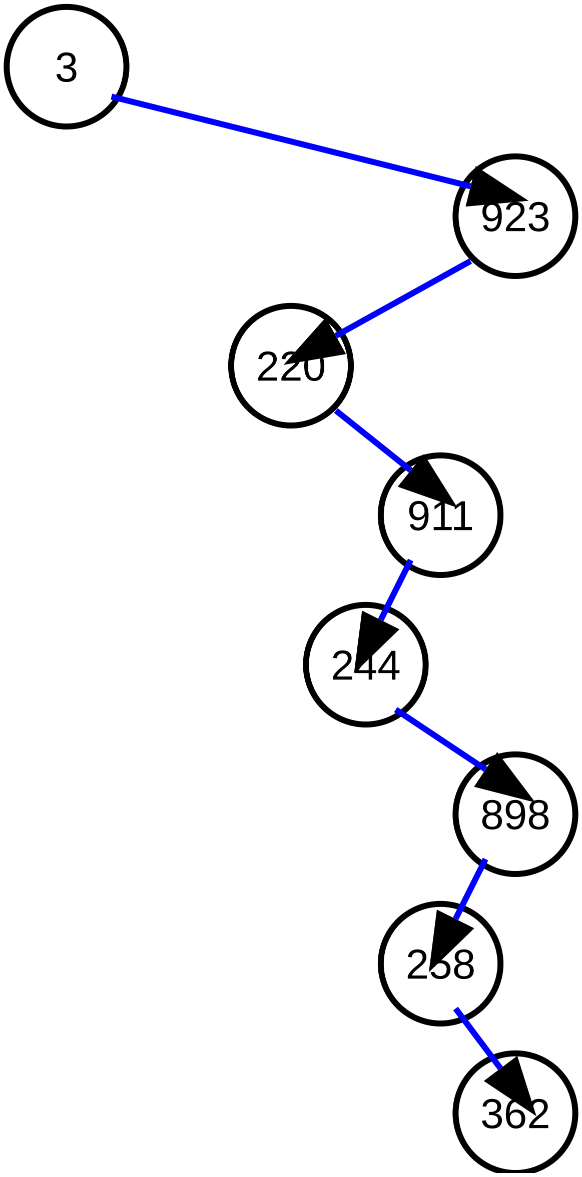
- After 381, inserting 278 violates the BST property because it suggests that 278 is smaller than 381 but is being considered **after** 381, which contradicts the binary search tree rules during search or insertion.

Conclusion:

The correct analysis shows that **Sequence (b)** `399 387 219 266 382 381 278 363` **cannot be the sequence of keys examined** because inserting 278 after 381 in a BST structure violates the BST properties. This confirms that sequence (b) is indeed the invalid one.

```
In [8]: show_svg("t3.svg")
```

Out[8]:



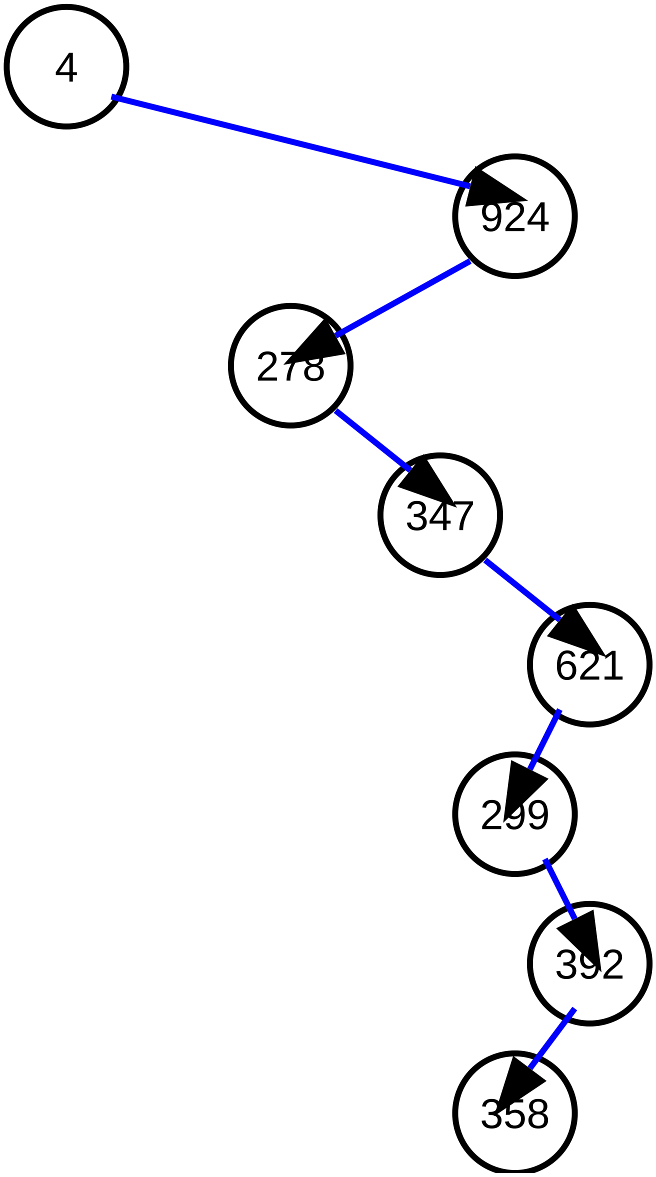
Sequence C: `3 923 220 911 244 898 258 362 363`

- 3 < 923 (valid move to the right)
- 923 > 220 (valid move to the left)
- 220 < 911 (valid move to the right)
- 911 > 244 (valid move to the left)
- 244 < 898 (valid move to the right)
- 898 > 258 (valid move to the left)
- 258 < 362 (valid move to the right)
- 362 < 363 (valid move to the right)

This sequence is valid.

In [9]:
```
show_svg("t4.svg")
```

Out[9]:



Sequence D: `4 924 278 347 621 299 392 358 363`

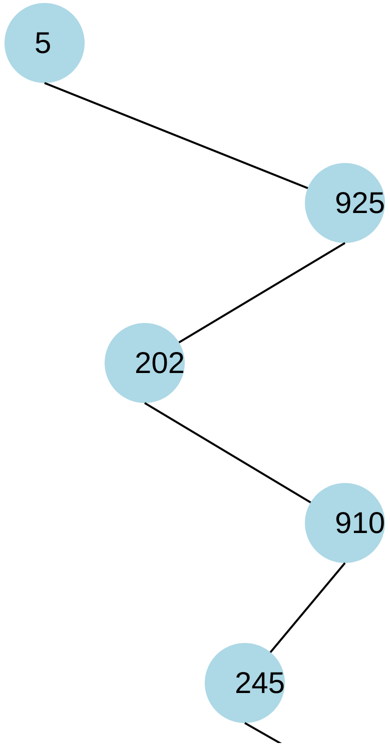
- 4 < 924 (valid move to the right)
- 924 > 278 (valid move to the left)
- 278 < 347 (valid move to the right)
- 347 < 621 (valid move to the right)
- 621 > 299 (valid move to the left)
- 299 < 392 (valid move to the right)
- 392 > 358 (valid move to the left)
- 358 < 363 (valid move to the right)

This sequence is valid.

In [10]:

```
show_svg("t5.svg")
```

Out[10]:



Sequence E: `5 925 202 910 245 363`

- 5 < 925 (valid move to the right)
- 925 > 202 (valid move to the left)
- 202 < 910 (valid move to the right)
- 910 > 245 (valid move to the left)
- 245 < 363 (valid move to the right)

This sequence is valid.